Other population-based metaheuristics

- IS - Immune Systems
- DE – Differential Evolution
- PMB - Probabilistic Model Building Algorithms
- MA – Memetic Algorithms

Immune Systems

Short history:

- mid 1980 - first models
- 1990 – Ishida proposes a first application of immune models in problem solving
- mid 1990:
  - Forrest et al: applications in computer security
  - Hunt et al: applications in data analysis
- Current tendency: back to the biological model

Applications

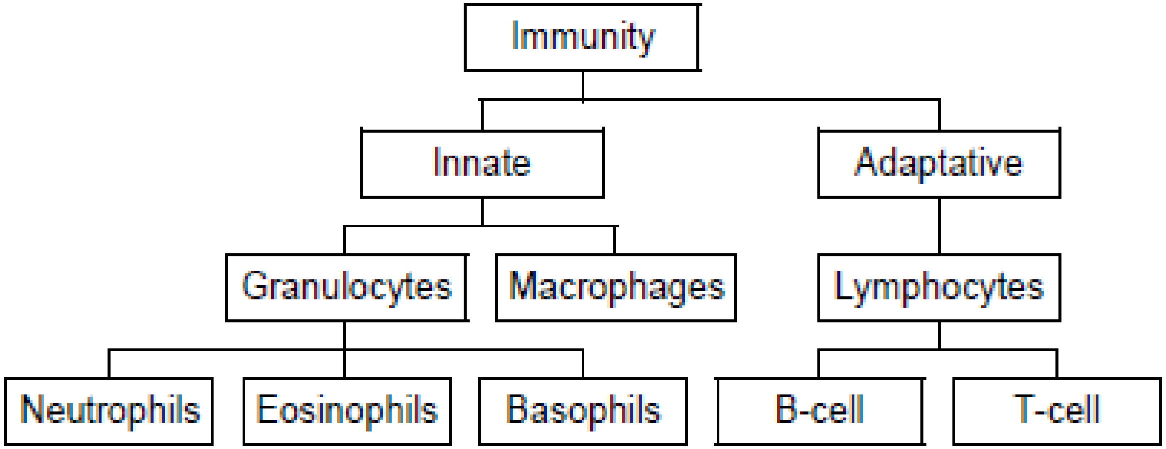
- Anomaly detection and information systems security
- Data analysis (classification, pattern recognition, clustering etc)
- Optimization;
- Self-organization and autonomous control;

Natural Immune Systems

The natural immune system contains two main components:

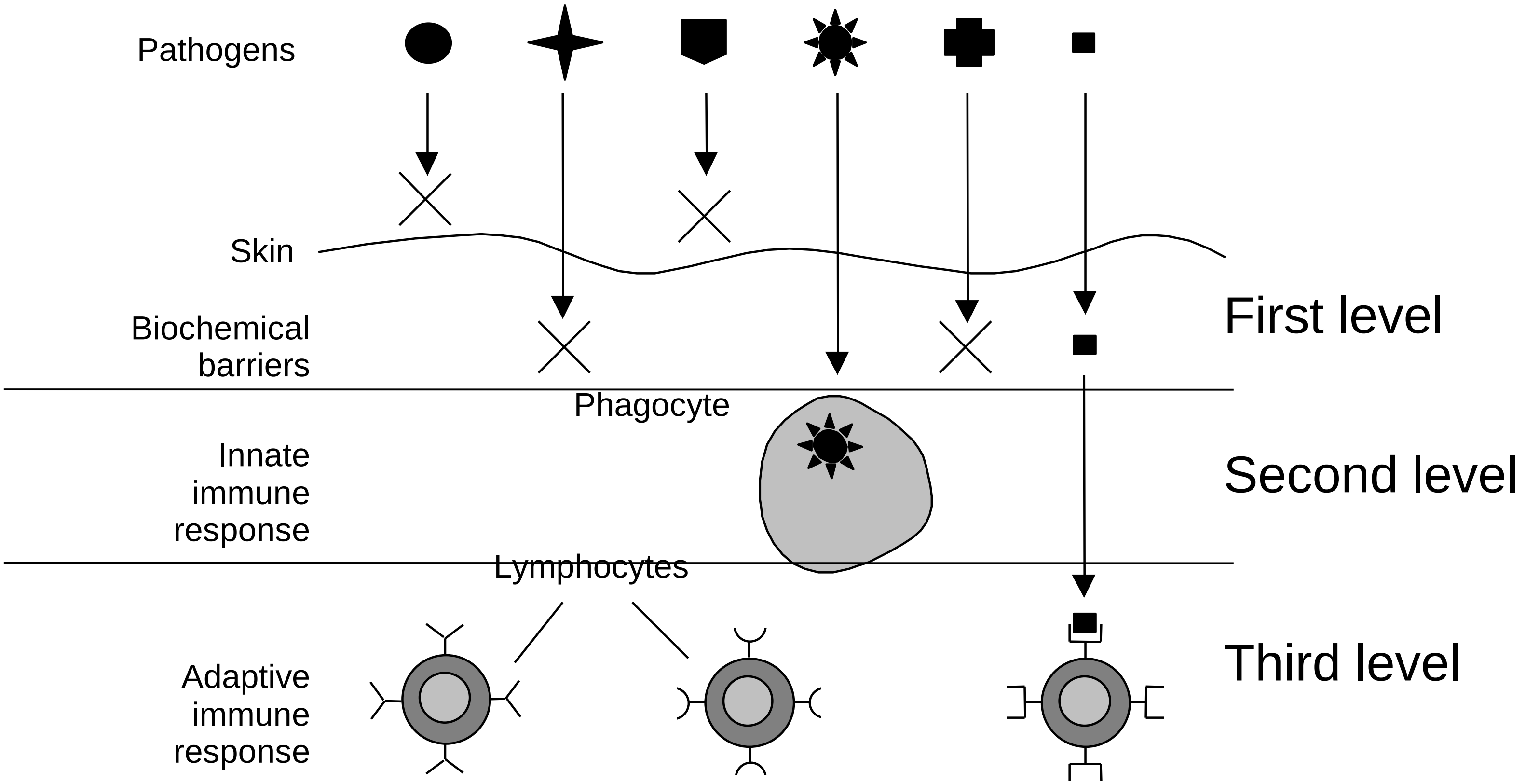
- innate (inherited from the parents) – based on granulocytes (neutrophils, eosinophils si basophils) and macrophages

- Adaptive – based on lymphocytes (B cells and T cells)



Natural immune system

Particularity: active at different levels



Natural immune systems

The adaptive component of the immune system is able to:

- Memorize (ability to recall previous contacts with pathogens and to react quickly)

- Learn (ability to identify/recognize unknown pathogens)

a) Active elements: lymphocytes

- They contain specific receptors able to recognize the antigens (the organisms usually contain a library of millions of receptors)
- There are two types of lymphocytes:
  - B cells
    - Synthesized in the bone marrow
    - Contain receptors called antibodies – the recognition process is based on the complementarity between the binding region of the B cell and the epitope of the antigen
  - T cells: Synthesized by thymus

Natural immune system

Main mechanisms

Negative selection: censoring the T cells which recognize the self components (they define the normal behaviour)

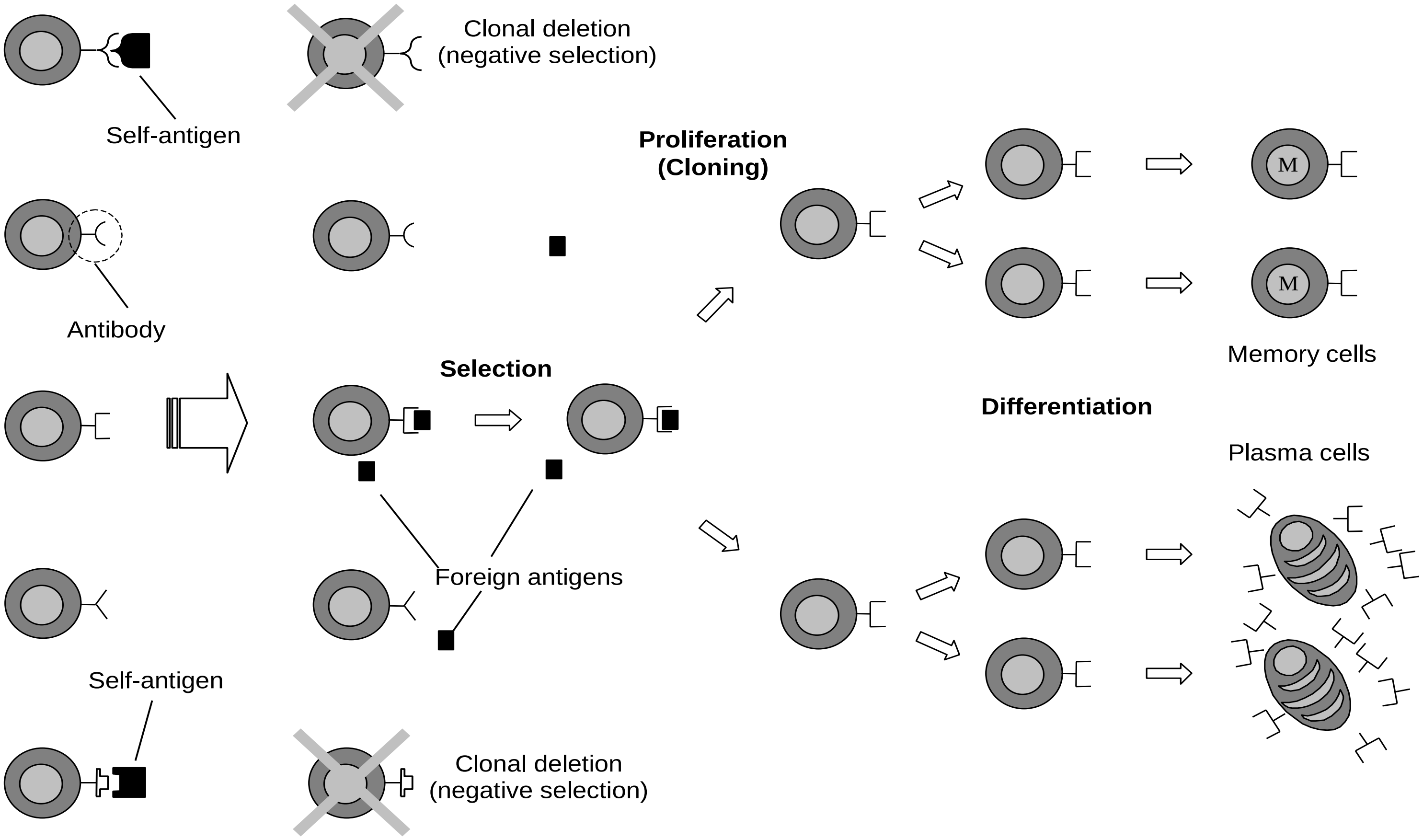
Clonal selection: proliferation and differentiation of cells which recognized an antigen (learning and generalization)

Affinity maturation: the affinity of B cells which recognized an antigen is reinforced by

- Mutation on the receptors (the mutation probability inversely correlated with the affinity)
- The storage of cells with high affinity in a memory (cells pool)
- Removal of the cells with incorrect behavior

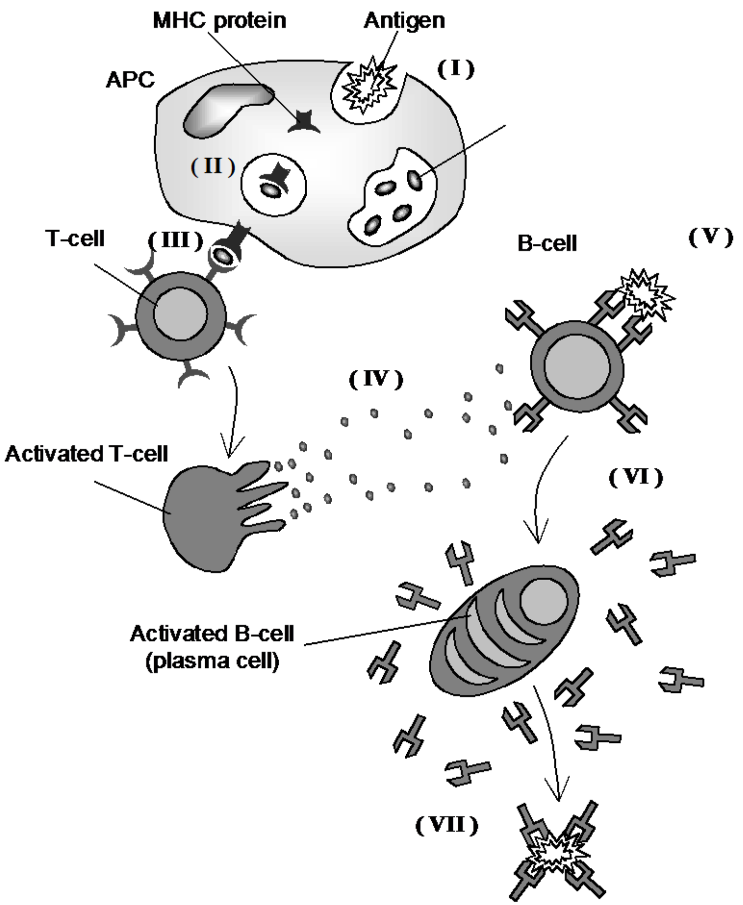
Natural immune system

Main mechanisms:



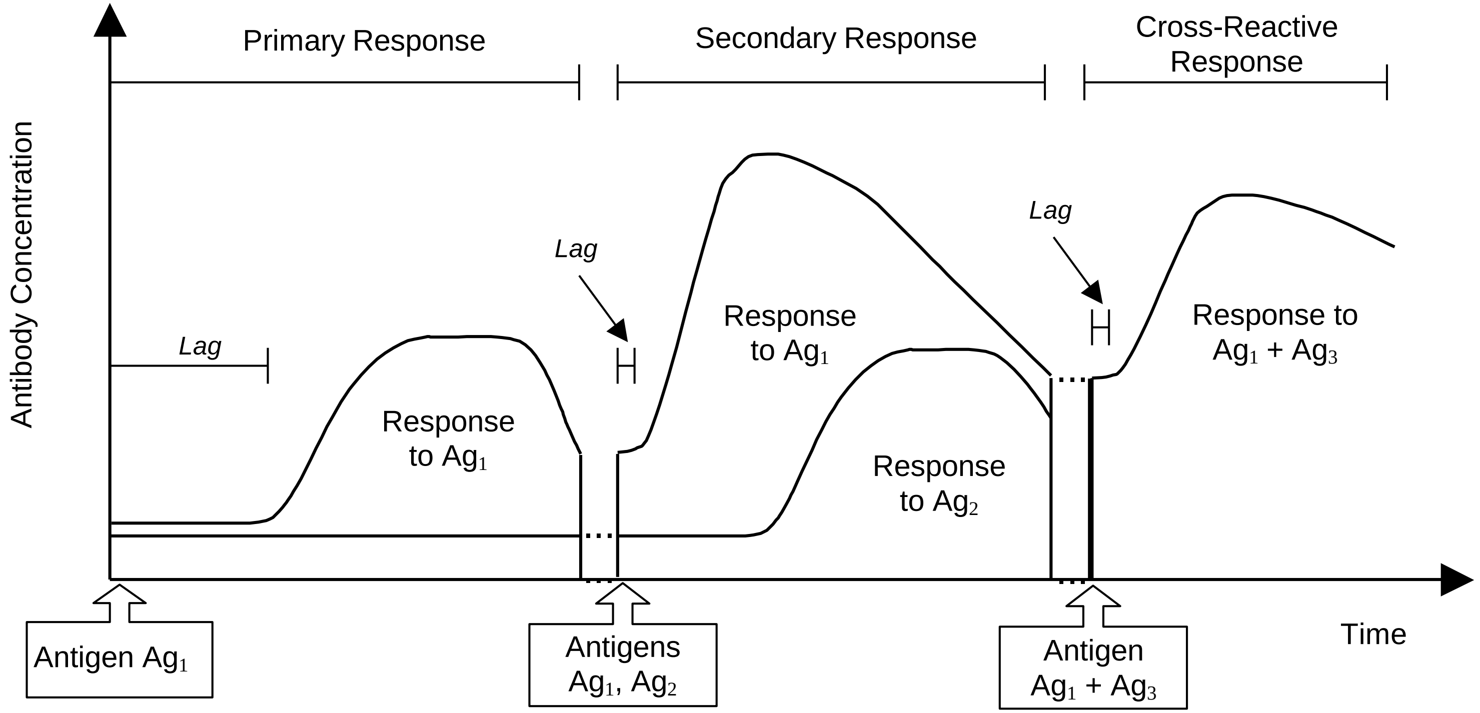
Natural immune system

Main steps:



Natural immune system

Primary and secondary reaction



Primary reaction: first answer at the contact with an antigen

Secondary reaction: rapid answer

AIS = Artificial Immune System

Idea of AIS based problem solving:

Problem to be solved = environment

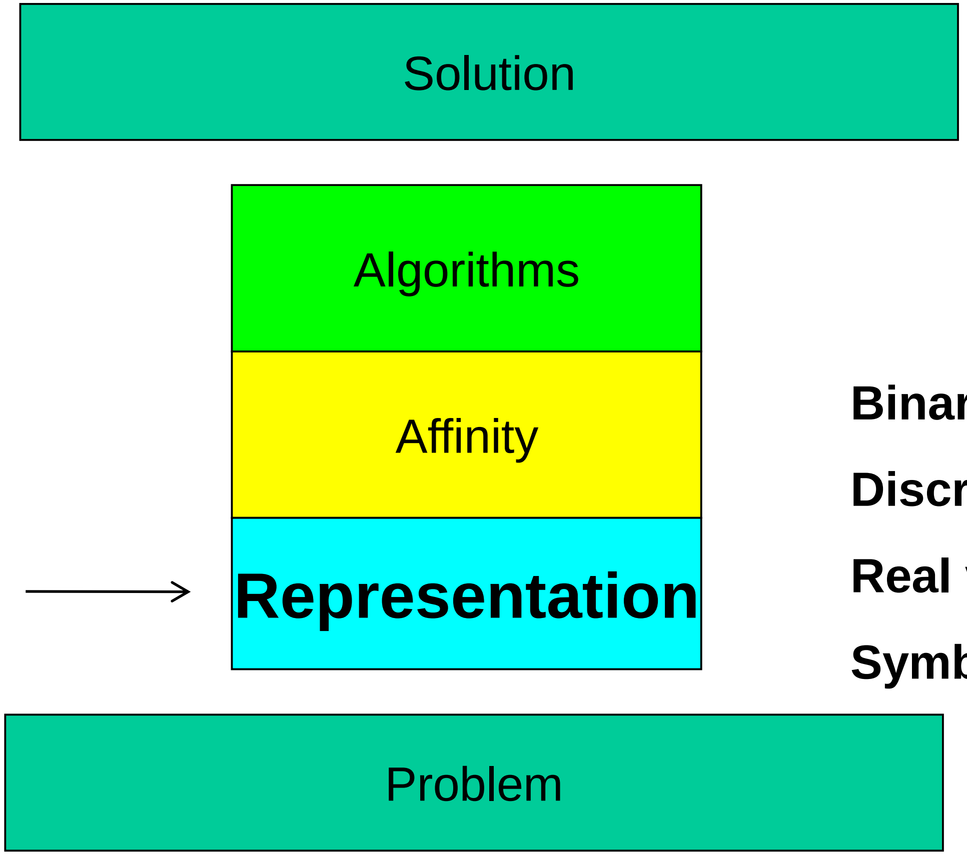
Solution (unknown) = antigen

Approximation of the solution (population element) = antibody

Measure of the quality of an element = affinity

AIS = Artificial Immune System

Main idea of AIS [DeCastro, Timmis, 2002]



Solution

Algorithms

Affinity

Representation

Problem

Binary values

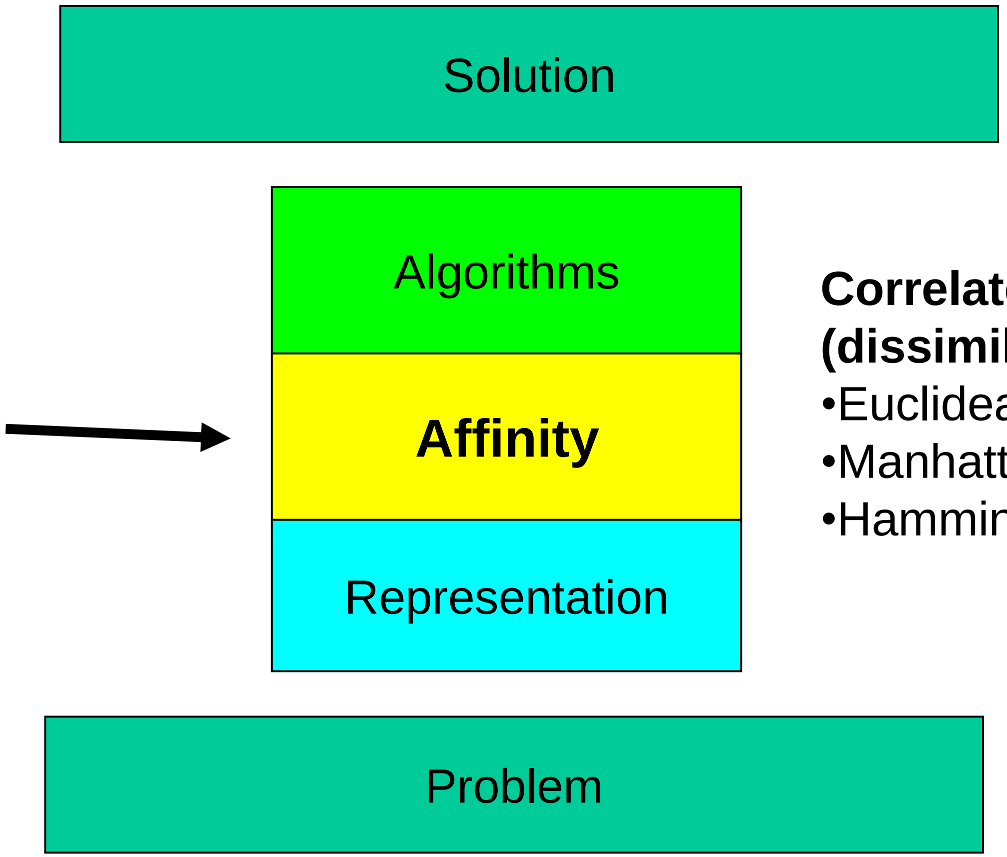
Discrete values

Real values

Symbolic values

AIS = Artificial Immune System

Main idea of AIS[DeCastro, Timmis, 2002]

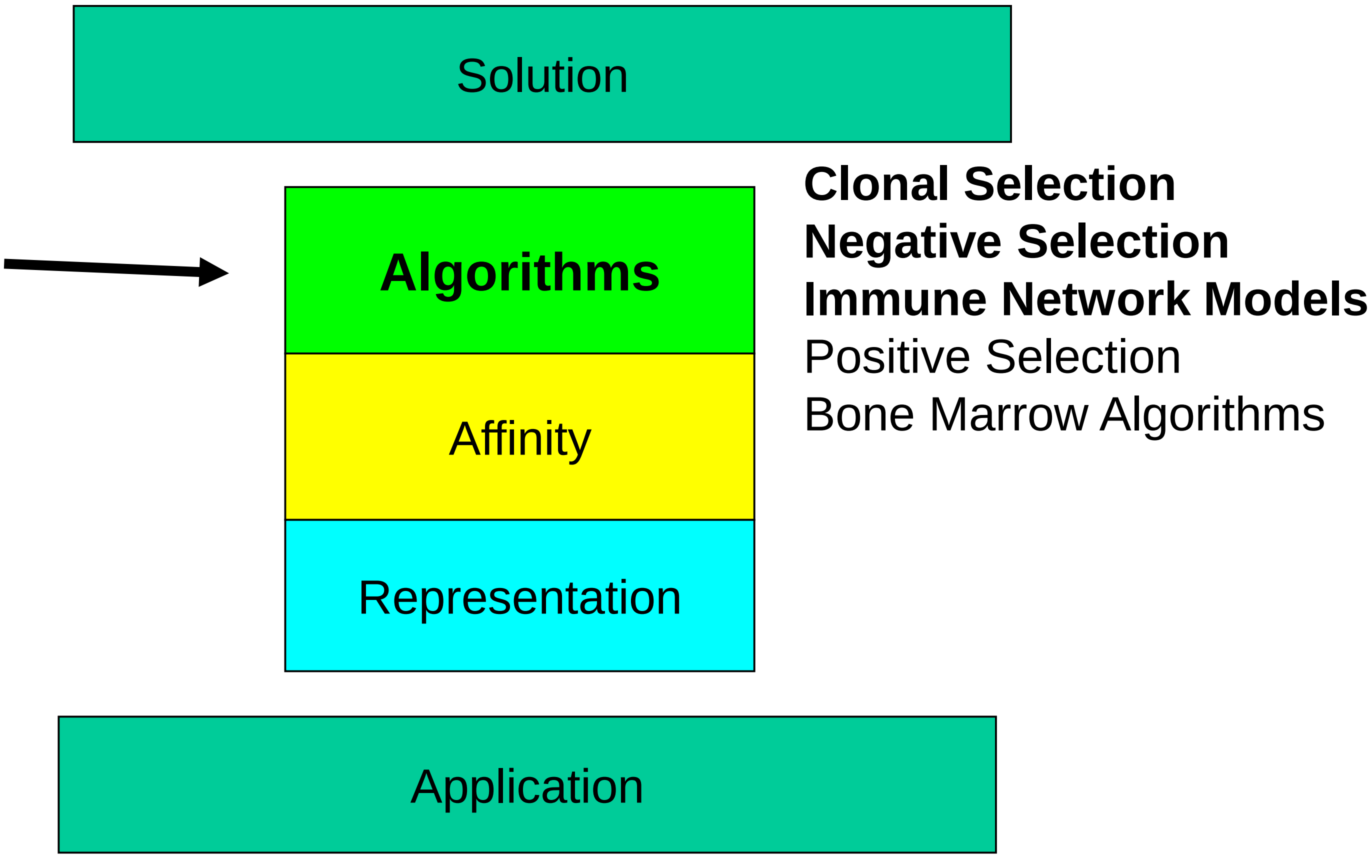


Correlated to a distance (dissimilarity)

- Euclidean
- Manhattan
- Hamming

AIS = Artificial Immune System

Main idea of AIS [DeCastro, Timmis, 2002]



Solution

Algorithms

Affinity

Representation

Application

Clonal Selection
Negative Selection
Immune Network Models
Positive Selection
Bone Marrow Algorithms

AIS = Artificial Immune System

CLONALG (Clonal Selection)

```
Initialization
REPEAT
        Antigenic presentation
        a.   Affinity evaluation
        b.   Clonal selection and expansion
        c.   Affinity maturation
        d.   Metadynamics
UNTIL "stopping condition"
```

AIS = Artificial Immune System

- CLONALG (Clonal Selection)

- Creates a population of antibodies

```
Initialization
REPEAT
  Antigenic presentation
     a.   Affinity evaluation
     b.   Clonal selection and expansion
     c.   Affinity maturation
     d.   Metadynamics
UNTIL "stopping condition"
```

AIS = Artificial Immune System

CLONALG (Clonal Selection)

```
Initialization
REPEAT
  Antigenic presentation
      a.    Affinity evaluation
      b.    Clonal selection and expansion
      c.    Affinity maturation
      d.    Metadynamics
UNTIL “stopping condition”
```

For each data (antigen) the steps 1-d are executed

AIS = Artificial Immune System

CLONALG (Clonal Selection)

```
Initialization
REPEAT
    Antigenic presentation
        a.    Affinity evaluation
        b.    Clonal selection and expansion
        c.    Affinity maturation
        d.    Metadynamics
UNTIL "stopping condition"
```

Compute the affinity

a) Data mining pb: affinity is higher if the similarity is higher

b) Optimization pb: affinity is higher if the fitness is higher (the fitness is correlated with the objective function value)

AIS = Artificial Immune System

CLONALG (Clonal Selection)

```
Initialization
REPEAT
  Antigenic presentation
    a.   Affinity evaluation
    b.   Clonal selection and expansion
    c.   Affinity maturation
    d.   Metadynamics
UNTIL "stopping condition"
```

- Select n elements from P in decreasing order of affinity
- Generate for each selected element a number (proportional to the affinity) of clones

AIS = Artificial Immune System

CLONALG (Clonal Selection)

```
Initialization
REPEAT
  Antigenic presentation
     a.   Affinity evaluation
     b.   Clonal selection and expansion
     c.   Affinity maturation
     d.   Metadynamics
UNTIL “stopping condition”
```

- Apply mutation to each clone
- The mutation rate is inverse proportional to the affinity
- Add the new element to the population
- Evaluate the affinity for new elements and store the best element

AIS = Artificial Immune System

- CLONALG (Clonal Selection)

```
Initialization
REPEAT
   Antigenic presentation
        a.   Affinity evaluation
        b.   Clonal selection and expansion
        c.   Affinity maturation
        d.   Metadynamics

UNTIL “stopping condition”
```

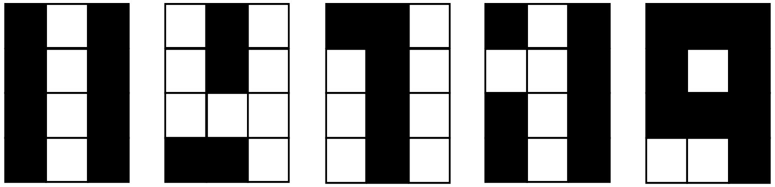
- Some of the elements of the population having small affinity are replaced with random elements

AIS = Artificial Immune System

Applications of CLONALG

- Pattern recognition = generate “detectors” for the recognition of characters specified by bitmaps

Rmk: affinity is measured using the Hamming distance

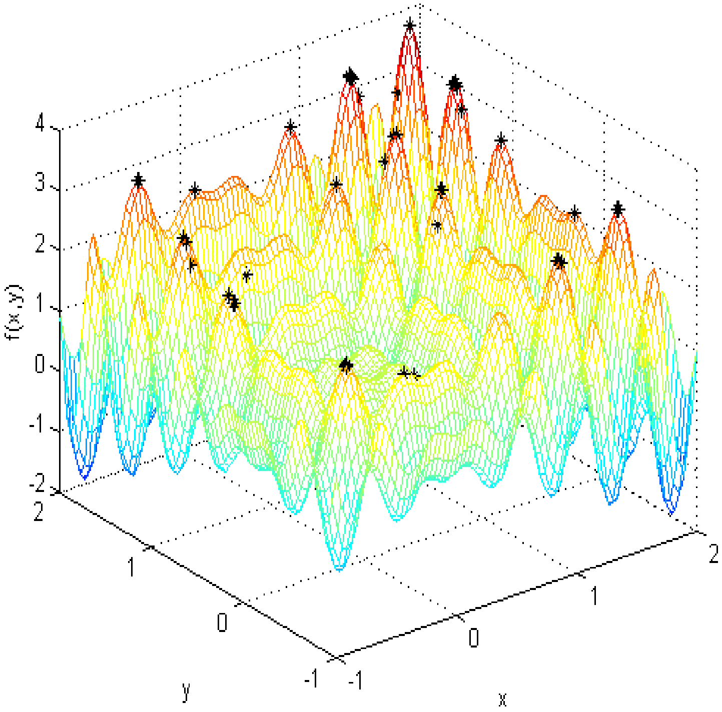


$$\mathbf{P} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

AIS = Artificial Immune System

Applications of CLONALG

- Multi-modal optimization = identify all optima (local and global) of a function



AIS = Artificial Immune System

Properties of CLONALG

- The general structure is similar to the structure of an evolutionary algorithm (instead of fitness is used the affinity)

- The specific elements refer to :
  - The cloning process is controlled by the value of the affinity
  - The mutation probability is inverse proportional to the affinity
  - The low affinity elements are replaced with random elements

AIS = Artificial Immune System

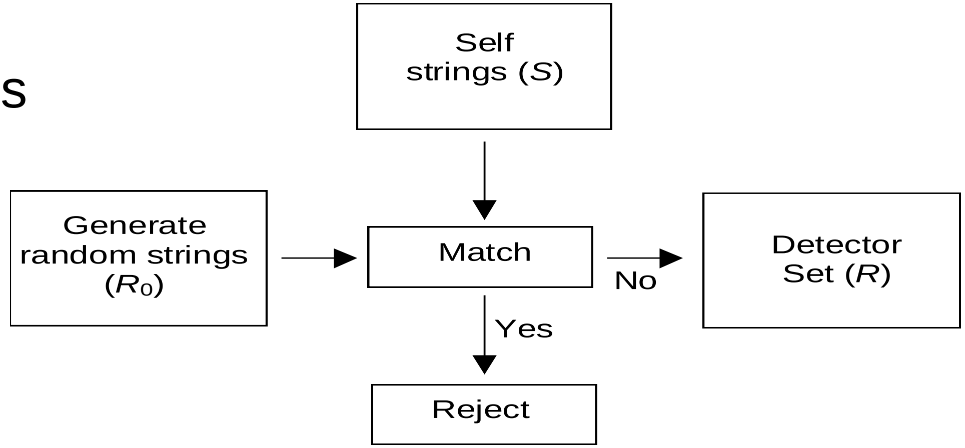
Negative selection algorithm

- It is based on the pronciple of the discrimination between self and non-self
- The self elements are considered to be representations of the normal behavior of a system
- The aim of the algorithm is to generate a set of detectors which are different from the set S of self elements (they would be detectors of non-self elements – would correspond to anomalous behavior)
- The algorithm will monitor the system functioning and will detect elements similar to non-self.

AIS = Artificial Immune System

Negative selection algorithm

Generating the set of detectors



Self strings (*S*)

Generate random strings (*R_0*)

Match

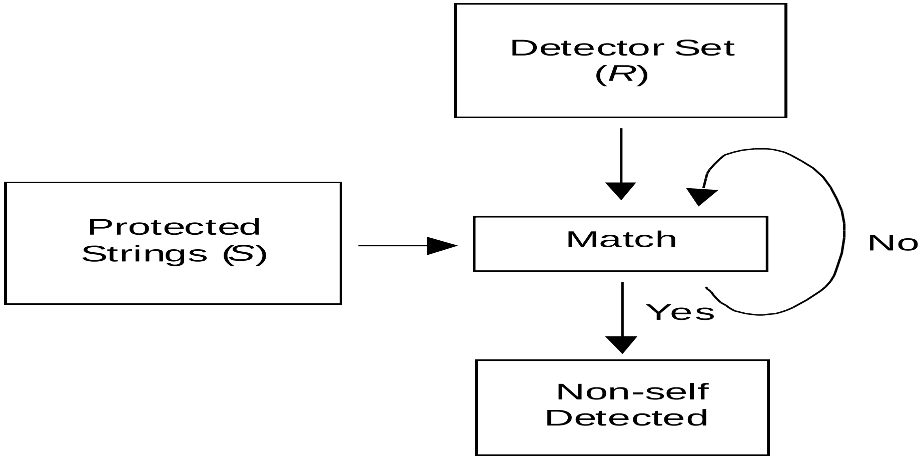
No

Detector Set (*R*)

Yes

Reject

System monitoring



Detector Set (*R*)

Protected Strings (*S*)

Match

No

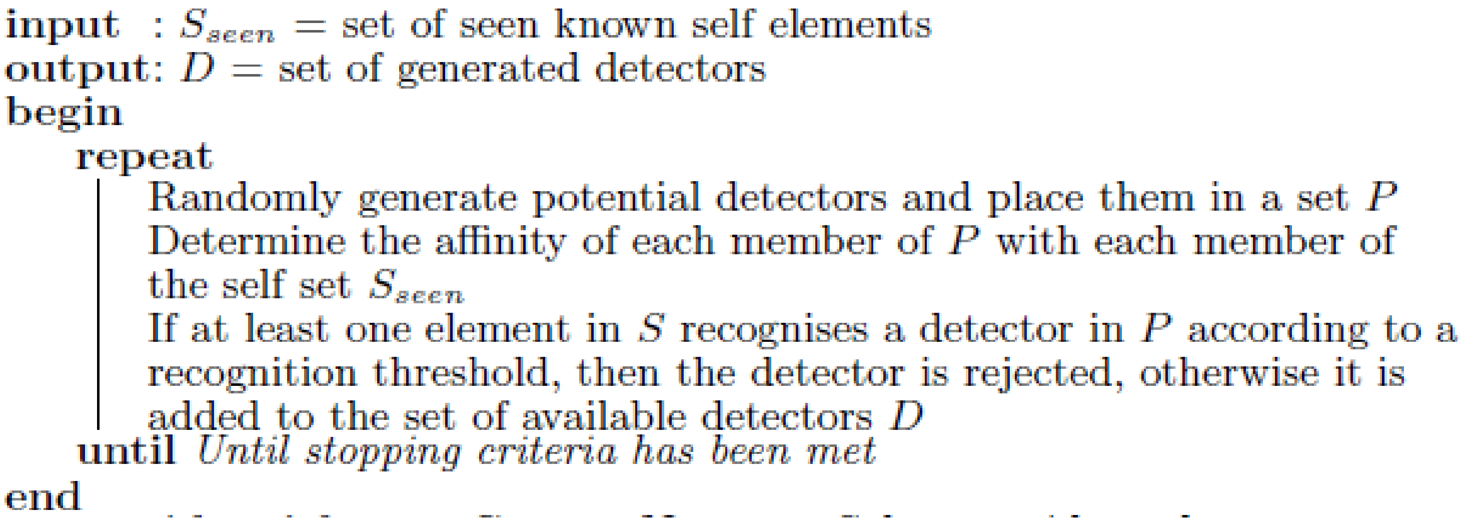
Yes

Non-self Detected

Applications: computer security (intruders detection) – limited applicability

AIS = Artificial Immune System

Negative selection algorithm



```
input  : $S_{seen}$ = set of seen known self elements
output: $D$ = set of generated detectors
begin
    repeat
    |   Randomly generate potential detectors and place them in a set $P$
    |   Determine the affinity of each member of $P$ with each member of
    |   the self set $S_{seen}$
    |   If at least one element in $S$ recognises a detector in $P$ according to a
    |   recognition threshold, then the detector is rejected, otherwise it is
    |   added to the set of available detectors $D$
    until Until stopping criteria has been met
end
```

J.Timmis, P. Andrews, N. Owens, E. Clark – An Interdisciplinary Perspective of Artificial Immune Systems, Evolutionary Intelligence, Volume 1, Number 1, 5-26, 2008

AIS = Artificial Immune System

aiNET Algorithm

```
Initialization
REPEAT
•   Antigenic presentation
    a. Affinity evaluation
    b. Clonal selection and expansion
    c. Affinity maturation
    d. Metadynamics
    e. Clonal suppression
•   Network interactions (analysis of interactions between network
    antibodies = computation of affinity between pairs of antibodies)
•   Network suppression (eliminate the antibodies which are similar to
    other antibodies)
•   Diversity (insertion of random antibodies)
UNTIL “stopping condition”
```

AIS = Artificial Immune System

Properties of aiNET:

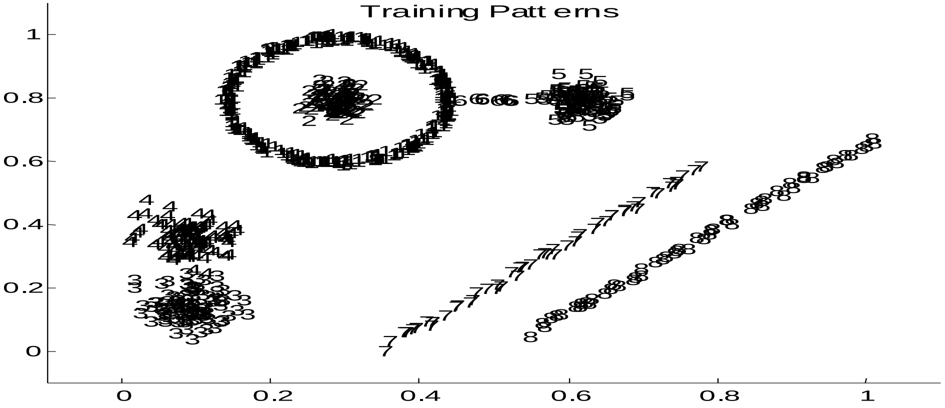
- aiNET is similar to CLONALG but it uses a suppression mechanism based on the affinity between the population elements
- aiNET was initially used for data clustering (but it has difficulty in the case of arbitrary distributed data)
- aiNET was successfully applied in solving multimodal optimization problems

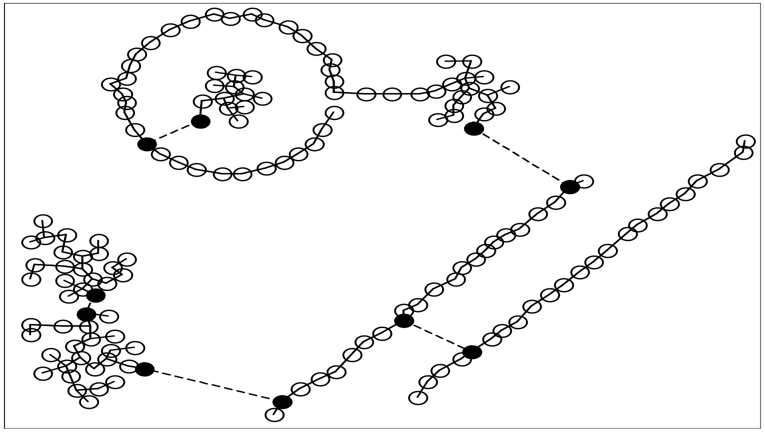
AIS = Artificial Immune System

aiNET - clustering

Training Pattern

Result immune network

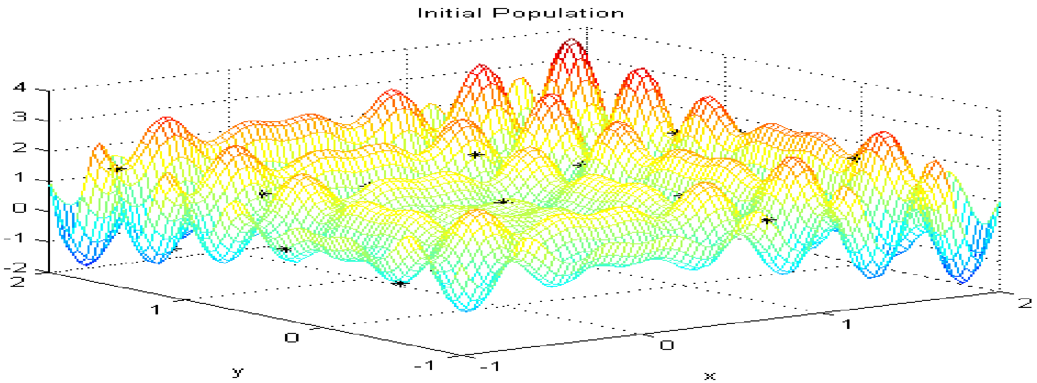




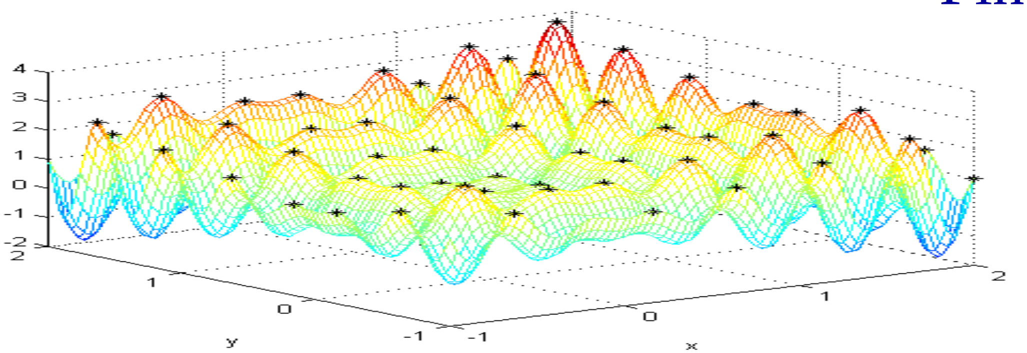
AIS = Artificial Immune System

aiNET - multimodal optimization

Initial population



Final population



Differential Evolution (DE)

Creators: Rainer Storn & Kenneth Price (1995)

Aim: continuous optimization

Idea: for each element of the current population:

- Randomly select 3 elements
- The mutation is based on the computation of the difference between two of the three selected elements; the difference (multiplied by a scale factor) is added to the third element. The obtained element is called mutant
- The mutant element is recombined with the current element leading to the so-called trial element
- If the trial element is better than the current element then it replaces it

Differential Evolution (DE)

Problem: maximization of f:D⊂R^n→R

$X = \{x_1, \ldots, x_m\}$ – current population

$Y = \{y_1, \ldots, y_m\}$ – population of candidates (trial vectors)

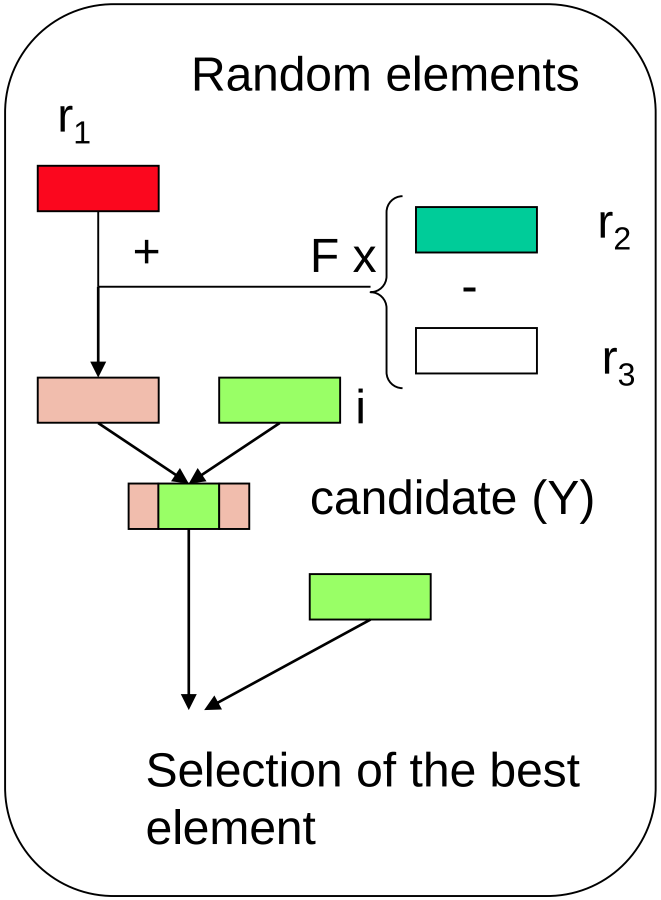
$Z = \{z_1, \ldots, z_m\}$ – new population

$$y_i^j = \begin{cases} x_{r_1}^j + F \cdot (x_{r_2}^j - x_{r_3}^j), & \text{with probability } p \\ x_i^j, & \text{with probability } 1-p \end{cases}$$

$r_1, r_2, r_3 =$ random indices from $\{1, \ldots, m\}$

$F \in (0,2],\ p \in (0,1]$

$$z_i = \begin{cases} x_i, & f(x_i) > f(y_i) \\ y_i, & f(x_i) \le f(y_i) \end{cases}$$



Differential Evolution (DE)

Variants

$$y_i^j = \begin{cases} x_{r_1}^j + F \cdot (x_{r_2}^j - x_{r_3}^j) \cdot N(0,1), & \text{with probability } p \\ x_i^j, & \text{with probability } 1-p \end{cases}$$

$$y_i^j = \begin{cases} x_{r_1}^j + F_1 \cdot (x_{r_2}^j - x_{r_3}^j) + F_2 \cdot (x_{r_4}^j - x_{r_5}^j), & \text{with probability } p \\ x_i^j, & \text{with probability } 1-p \end{cases}$$

$$y_i^j = \begin{cases} \lambda x_*^j + (1-\lambda) x_{r_1}^j + F \cdot (x_{r_2}^j - x_{r_3}^j), & \text{with probability } p \\ x_i^j, & \text{with probability } 1-p \end{cases}$$

$x_* = \text{best element of the population}$

Taxonomy: DE/base element/number of differences/crossover type

(e.g. DE/rand/1/bin, DE/rand/2/bin, DE/best/1/bin etc.)

Differential Evolution (DE)

Control parameters:

Scale factor (F):

- range: (0,2)
- small values: exploitation of the search space (local search)
  can lead to premature convergence
- large values: exploration of the search space

Typical value: F=0.5

Crossover probability:

- small values (<0.5): appropriate for separable problems
- large values (>0.5): appropriate for nonseparable problems

Typical value: p=0.9 (rmk: the crossover probability is frequently denoted by CR)

Differential Evolution (DE)

Adaptive DE = the parameters F and p are modified during the generations based on:

- Historical records of successful trial elements - in this case each element in the population has its own values for F and p
- Random perturbation
- Selection out of a pool of possible values

Examples:

- jDE (J. Brest, 2006)
- JADE (Zhang & Sanderson, 2009)
- SHADE (Tanabe & Fukunaga, 2013)

Differential Evolution (DE)

jDE overview:

- each element (x_i) in the population has its own values for F and p (F_i and p_i)
- If a trial element is successful (better than the parent) its parameters are preserved, otherwise they are randomly changed:

$$F_i^{new} = \begin{cases} rand(0.1,1) & \text{if } rand(0,1) < \tau_F \\ F_i^{old} & \text{otherwise} \end{cases}$$

$$p_i^{new} = \begin{cases} rand(0,1) & \text{if } rand(0,1) < \tau_p \\ p_i^{old} & \text{otherwise} \end{cases}$$

Initial values: F=0.5, p=0.9

New parameters: τ_F and τ_p

Differential Evolution (DE)

JADE overview:

- Mutation variant: DE/p-best = the base vector is randomly selected out of the best p% elements of the population

- The elements from a population which are replaced by better children are stored in an archive and the element which is subtracted (x_{r3}) is selected from this archive

- At each generation the parameters corresponding to each element are resampled using some probability distributions:
  - F_i is generated using Cauchy(mF_i,0.1)
  - p_i is generated using Normal(mp_i,0.1)
  - mF and mp are initialized with 0.5 (for all elements) and are computed based on averages of parameters corresponding to successful trial elements

Differential Evolution (DE)

SHADE overview:

- Similar with JADE
- Main difference: the adaptation rule of parameters m_F and m_p: instead of using the average of values in the archive with successful parameters a random value is selected from an archive which contain the averages of the distributions used to generate successful parameters

Probabilistic Model Building Algorithms

Particularity: class of algorithms which search the solution space by estimating and simulating some probability distributions

Variants:

- Estimation of Distribution Algorithms (EDA) [Mühlenbein & Paass, 1996]
- Iterated Density Estimation Algorithms (IDEA) [Bosman & Thierens, 2000]
- Bayesian Optimization Algorithms (BOA) [Pelikan, Goldberg, & Cantu-Paz, 1998]

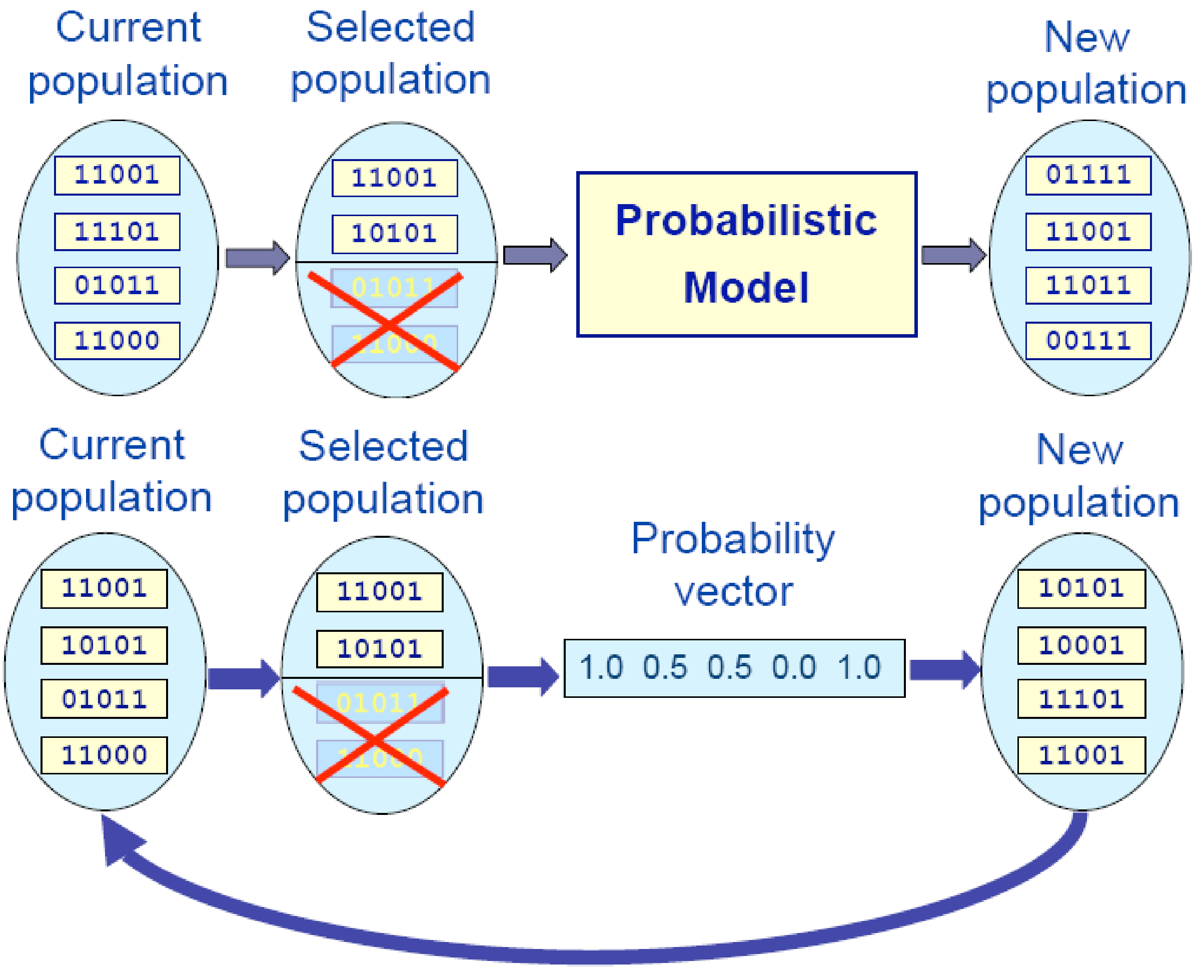
Idea: the mutation and crossover operators are replaced with

- a process for the estimation of the probability distribution of selected elements
- and a process of sampling new elements using this distribution

Remark: the sampled values should be promising elements

Probabilistic Model Building Algorithms

Illustration [M.Pelikan – Probabilistic Model Building GA Tutorial]



Probabilistic Model Building Algorithms

General structure.

Step 1: Population initialization (m elements)

Step 2: REPEAT

- select m'<m elements from the current population (based on their fitness)
- estimate a probability distribution using the selected elements
- sample m elements from the estimated probability distribution

UNTIL <stopping condition>

Probabilistic Model Building Algorithms

Remarks

- The main difficulty is to estimate the probability distribution (especially when the components of individuals are correlated)

- A simplified variant is based on the assumption that the components are independent; therefore the corresponding probabilities can be estimated separately.

Variants based on the independence assumption:

- UMDA (Univariate Marginal Distribution Algorithm)
- PBIL (Probabilistic Based Incremental Learning)

Probabilistic Model Building Algorithms

UMDA (Mühlenbein, Paass, 1996)

$$P^t(x_i) = \frac{\sum_{j=1}^{m'} \delta_j(X_i = x_i \mid S(t-1))}{m'} \quad \text{probability of component i}$$

$S(t\text{-}1)$ is the population selected at iteration $(t\text{-}1)$

$\delta_j(X_i = x_i \mid S(t-1)) = 1$ if the jth selected element contains the value x_i on position i

PBIL (Baluja, 1995)

$$P^t(x_i) = (1-\alpha)P^{(t-1)}(x_i) + \alpha \frac{\sum_{j=1}^{m'} \delta_j(X_i = x_i \mid S(t-1))}{m'}$$

$\alpha \in (0,1]$

Memetic Algorithms

Creator: Pablo Moscato (1989)

Particularity: hybridization of EAs with local search techniques

Name: “memetic” comes “meme”, a term coined by Richard Dawkins to specify the transfer unit of different entities (biological, cultural etc) between generations

Variants: Hybrid Evolutionary Algorithms, Baldwinian Evolutionary Algorithms, Lamarckian Evolutionary Algorithms, Cultural Algorithms or Genetic Local Search

Memetic Algorithms

General structure:

Step 1: Population Initialization

Step 2: WHILE <stopping condition>

- Evaluate the elements of the population
- Generate new elements using the variation operators (mutation and crossover)
- Select a subpopulation on which are applied some local search operators (e.g. SA, TS etc)

Remarks:

1. The local search can be based on a set of operators – the operators to be applied are probabilistically selected
2. The elements which define the local search operators can be evolved.